

FIG. 1
PRIOR ART

1 c:\collections
2 notes.txt
3 myletter.doc
4 c-myhomepage
5
6 s
7 homepage.html
8 myphoto.jpg

FIG. 2

1 c:\collections
2 notes.txt
3 myletter.doc
4 c-myhomepage
5 cspec
6 s
7 homepage.html
8 myphoto.jpg

100

FIG. 3

1 collection c-myhomepage
2 coll-type cf-web-page
3 coll-desc A sample homepage collection
4 end-collection

102

FIG. 4

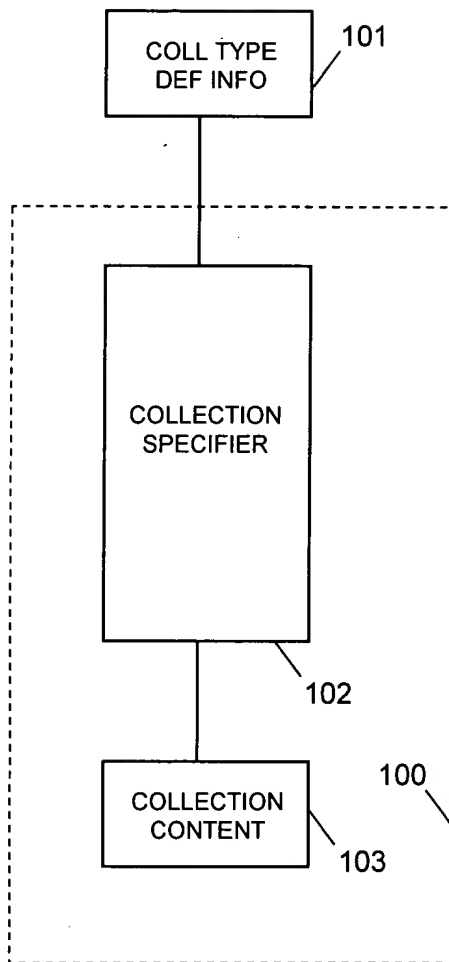


FIG. 5

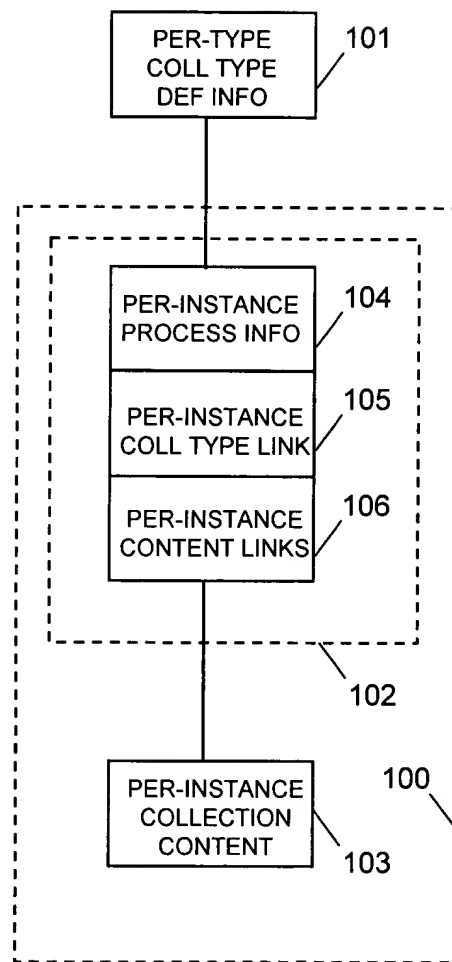


FIG. 6

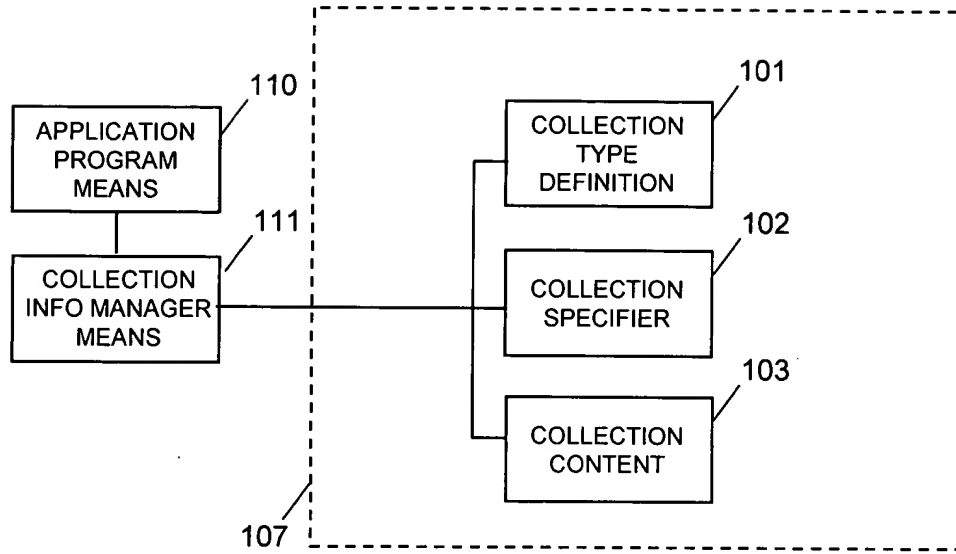


FIG. 7

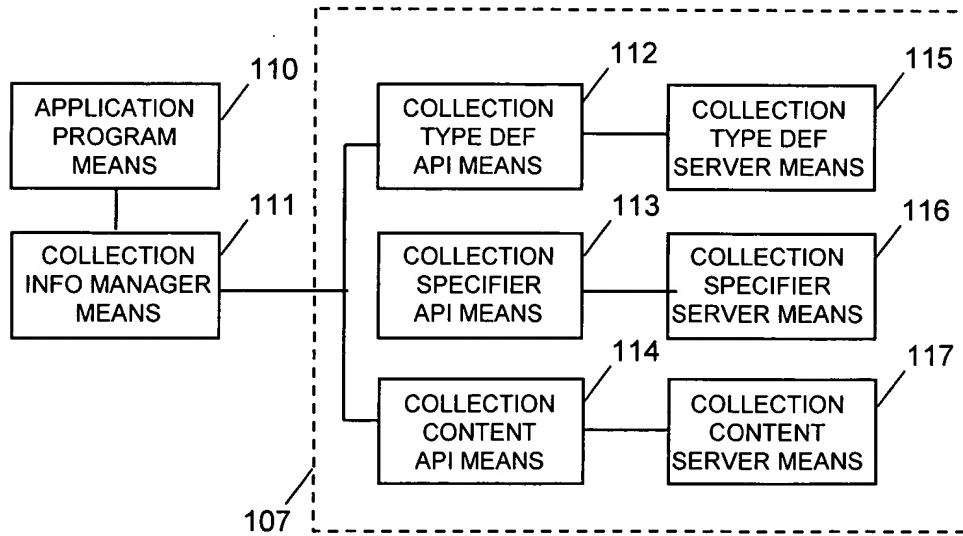


FIG. 8

```

1  /* collection data structure */
2  collection-info {

3      + specifier_info
4          + coll-type-indicator
5          + other specifier information ...

6      + content_info
7          + content_location_info ...
8          + content_members ...
9          + other content information...

10     + other collection structure information...
11 }

```

FIG. 9

```

1  /* collection type definition data structure */
2  collection-type-definition-info {

3      + coll-type-name
4      + collection internal structure info ...
5      + collection content location info ...
6      + collection content type recognition info ...

7      + other collection type definition information...
8  }

```

FIG. 10

<u>KEY</u>	<u>VALUE</u>
1 /* collection type internal structure definitions */	
2 dir_source_files	./s
3 dir_doc_files	./doc
4 /* content location definitions (per-type content links) */	
5 content_subtree_http	http://host.com/some/dir/name
6 content_subtree_ftp	ftp://host.com/some/dir/name
7 content_subtree_nfs	/some/local/directory/name
8 /* content type recognition definitions */	
9 content_policy	subtree_below_cspec_file
10 content_file_type	.c file_cpp
11 content_file_type	.c file_c
12 content_file_type	.h file_c_include
13 content_file_type	.doc file_ms_word
14 content_file_type	.html file_html
15 content_file_type	.xls file_ms_excel
16 /* collection processing definitions */	
17 compile_c_files	yes
18 compiler_windows	vc++
19 compiler_unix	gcc
20 build platforms	Win98, Win2000, gnulinux
21 process files	compile link
22 link libraries	stdio math sock
23 /* results dispatching definitions */	
24 results_ftp_host	ftp.output.com
25 results_ftp_dir	c:\ftphome\collection\results

DISCUSSION



```
1  /* simplified algorithm for collection makefile generator */
2  Call get runtime info to get invocation parameters
3  Call collection content classifier to classify collection content
4  Call collection makefile generator manager to generate
   a complete makefile, passing classifier information as input
```

7/41

FIG. 13

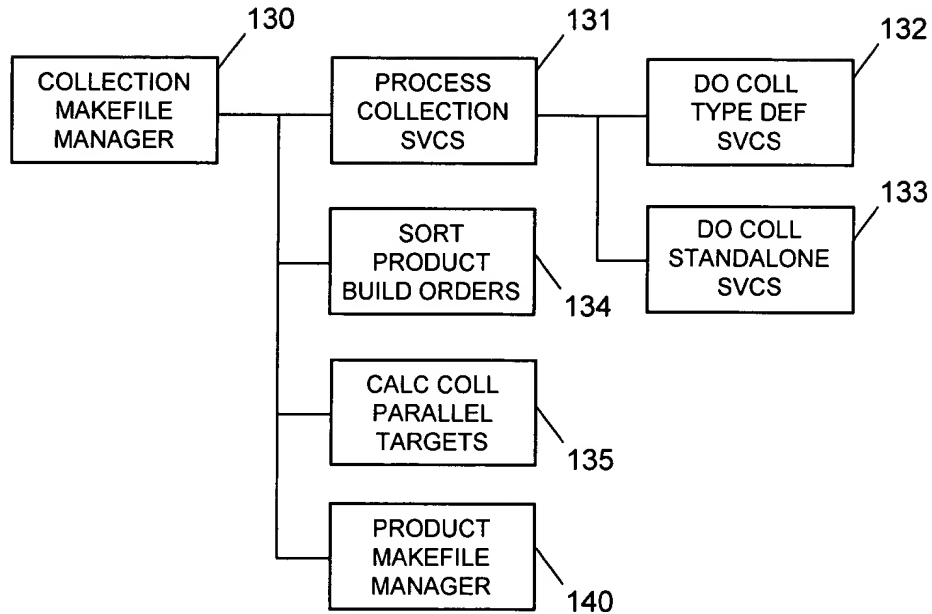


FIG. 14

- 1 /* simplified algorithm for collection makefile manager */
- 2 Process collection-level fragments
- 3 Process fragments from collection type definition
- 4 Process fragments from collection specifier
- 5 Determine relative build order among multiple products
- 6 Determine number, names of coll-level parallel build targets
- 7 Loop over each product in collection
- 8 Process each product by calling product makefile manager

FIG. 13

FIG. 15

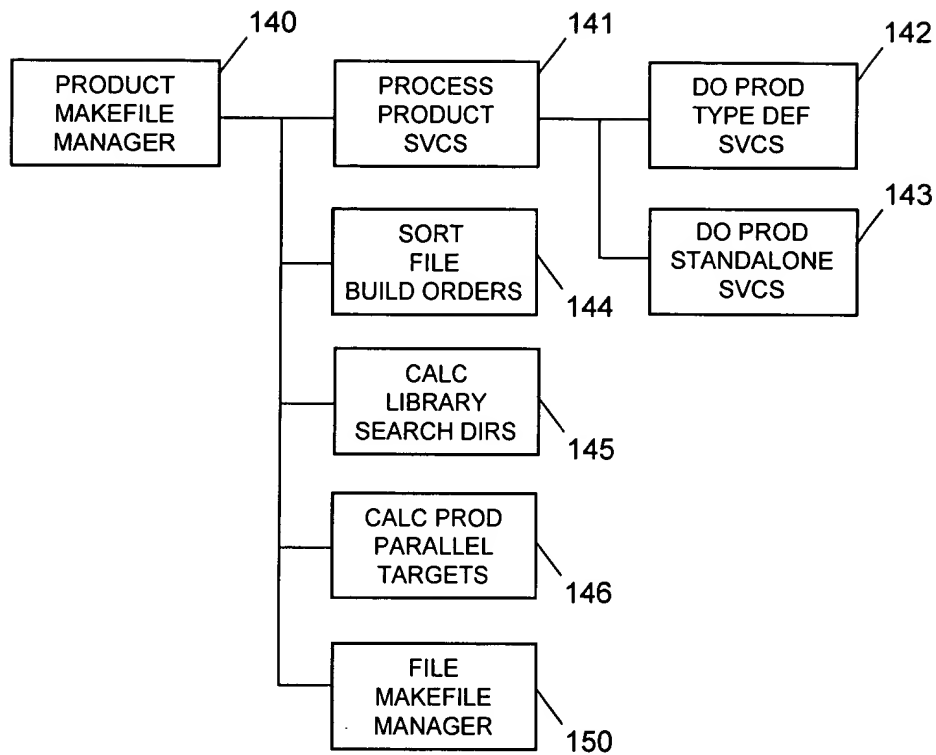


FIG. 16

- 1 /* simplified algorithm for processing one product */
- 2 Process product-level fragments
- 3 Process fragments from product type definition
- 4 Process fragments from product section of collection specifier
- 5 Determine relative build order among content files for product
- 6 Determine number, names of product-level parallel build targets
- 7 Loop over each content file
- 8 Process each content file by calling file makefile manager

FIG. 17

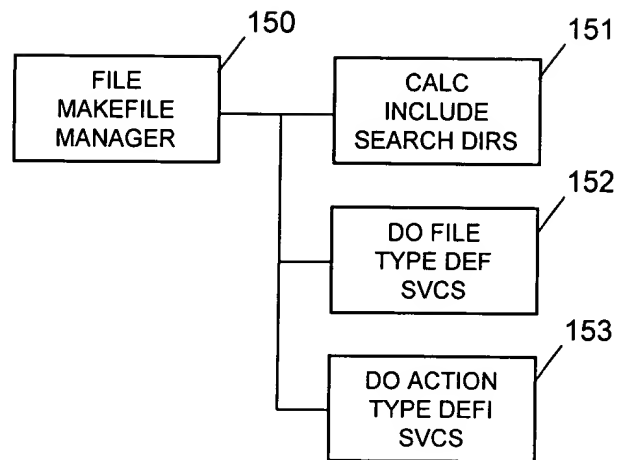


FIG. 18

- ```
1 /* simplified algorithm for processing one content file */
2 Calculate include file search directories
3 Process fragments from content type definition
4 Process fragments from action type definition
```

10/41

FIG. 19

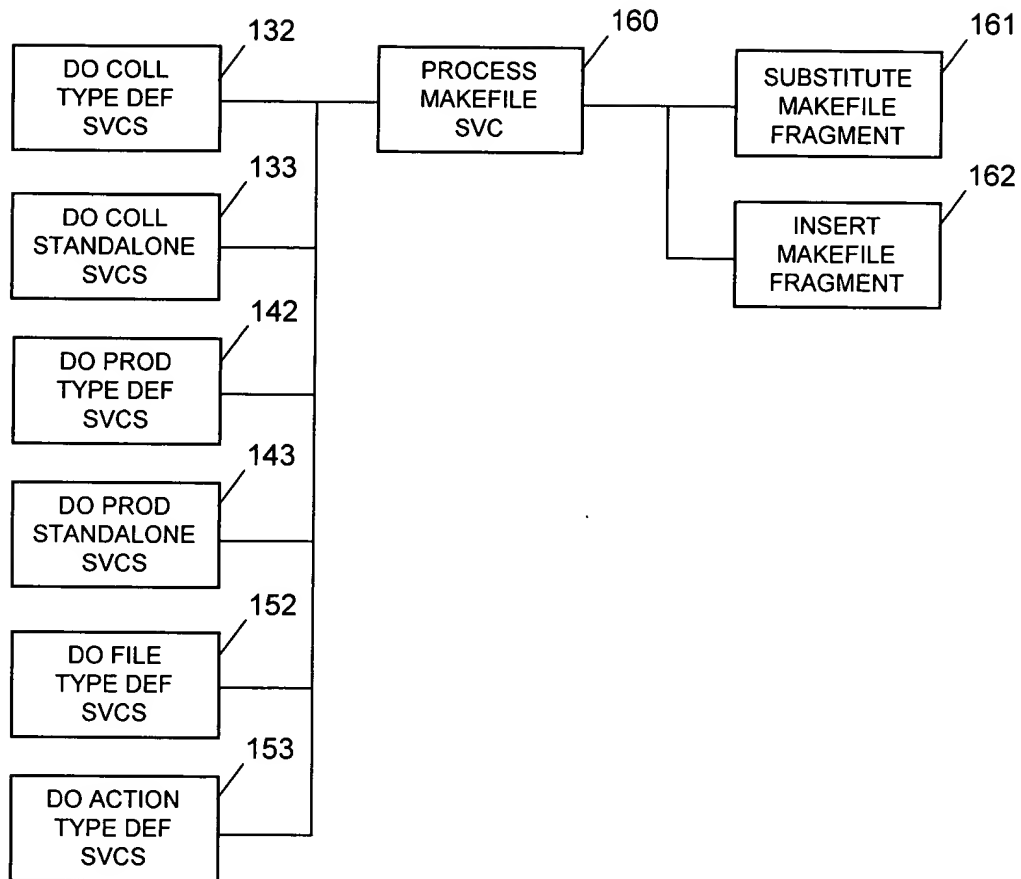


FIG. 20

- 1 /\* simplified algorithm for processing one fragment \*/
- 2 Substitute replacement values for placeholder strings
- 3 Insert substituted fragment into makefile data structure

11/41

FIG. 21

```
1 c:\collections
2 c-my-example
3 cspec
4 s
5 pi
6 cmdline.h
7 win98
8 cmdline.c
9 gnulinux2
10 cmdline.c
11 lib
12 pi
13 libfuncs.h
14 libfuncs.c
```

FIG. 22

```
1 cspec:

2 collection c-my-example
3 coll-type ct-program
4 coll-desc A multi-platform C program with library.
5 end-collection

6 product myprog
7 prod-type pt-program
8 prod-desc A program product.
9 libs team-lib gnulinux-lib
10 end-product

11 product mylibrary
12 prod-type pt-library
13 prod-desc A library product.
14 end-product
```

FIG. 23

```

1 /* classification output for gnulinux2 platform */
2 collection c-my-example
3 coll-type ct-program
4 ... other coll classification info

5 /* classification info for a program product */
6 product myprog
7 prod-type pt-program

8 content cmdline.h
9 content-path ../s/pi/cmdline.h
10 content-type ctype-c-header
11 content-language c
12 end-content

13 content cmdline.c
14 content-path ../s/gnulinux2/cmdline.c
15 content-type ctype-c-source
16 content-language c
17 content-dep ../s/pi/cmdline.h
18 content-dep ../lib/pi/libfuns.h
19 content-dep external-incl-file.h
20 content-dep team-incl.h
21 end-content

22 end-product

```

FIG. 24

```

1 /* classification output */
2 collection c-my-example

3 ... /* classification info for the host collection */
4 ... /* classification info for the program product */

5 /* classification info for a library product */
6 product mylibrary
7 prod-type pt-library

10 content libfuncs.h
11 content-path ../lib/pi/libfuncs.h
12 content-type ctype-c-header
13 content-language c
15 end-content

16 content libfuncs.c
17 content-path ../lib/pi/cmdline.c
18 content-type ctype-c-source
19 content-language c
21 content-dep ../lib/pi/libfuncs.h
22 end-content

23 end-product

```

FIG. 25

- 1 collection type definition information
- 2 product type definition information
- 3 content type definition information
- 4 action type definition information
- 5 cspec:
- 6 coll-type                      ct-program
- 7 index-coll-types.tbl:
- 8 ct-program                      ct-program.def
- 9 ct-web-page                      ct-web-page.def
- 10 ct-program.def:
- 11 product-type-index              index-product-types.tbl
- 12 index-product-types.tbl:
- 13 pt-program                      pt-program.def
- 14 pt-program.def:
- 15 content-type-index              index-content-types.tbl
- 16 index-content-types.tbl
- 17 ctype-c-source                      content-c.def
- 18 content-c.def:
- 19 action-type-index              index-action-types.tbl
- 20 index-action-types.def:
- 21 action-c-source                      action-c-source.def
- 22 action-c-source.def:
- 23 ... action definition information

FIG. 25

+

1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
17

```

1 index-prod-program.tbl:
2 pt-program pt-program.def
3 pt-program-java pt-program-java.def
4 pt-program-unix pt-program-unix.def
5 pt-program-win pt-program-win.def
6
7 pt-program.def:
8 /* type definition info for a "program" product type */
9 dir-source-files dirs-source.lst
10 dir-library-files dirs-library.lst
11 dir-include-files dirs-include.lst
12 file-identification-table file-identification.tbl
13 content-type-index index-content-types.tbl
14 service svc-prod-program
15 ... other product type info

```



FIG. 28

```

1 index-content-types.tbl:
2 ctype-c-source content-c.def
3 ctype-c-header content-c-h.def
4 ctype-csh content-csh.def
5 ctype-html content-html.def

6 content-c.def:
7 /* type definition info for a "c" file type */
8 type c-source
9 language c
10 action action-c-source
11 action-type-index index-action-types.tbl
12 service svc-file-c-source
13 ... other content type definition info

```

FIG. 29

```

1 index-action-types.tbl:
2 action-c-source action-c-source.def
3 action-c-header action-c-header.def
4 action-csh action-csh.def
5 action-html action-html.def

6 action-c-source.def:
7 parser-type internal
8 parser-name internal-c
9 service svc-action-c-source

```

FIG. 30

```
1 idx-makefile-services.tbl:
2 /* services for collections */
3 svc-coll-macro-platform coll-macro-platform.tpl
4 svc-coll-macro-site coll-macro-site.tpl
5 svc-coll-macro-compiler coll-macro-compiler.tpl
6 svc-coll-macro-toolnames coll-macro-toolnames.tpl
7 svc-coll-macro-file-suffix coll-macro-file-suffix.tpl
8 svc-coll-target-defaults coll-target-defaults.tpl
9 ...
10 /* services for products */
11 svc-prod-program prod_prog_pi.tpl
12 svc-prod-program prod_prog_os.tpl
13 svc-prod-program prod_prog_pd.tpl
14 ...
15 svc-prod-library prod-lib-pi.tpl
16 svc-prod-library prod-lib-os.tpl
17 svc-prod-library prod-lib-pd.tpl
18 ...
19 /* services for files */
20 svc-file-c-source file-c.tpl
21 svc-file-c-header file-c-header.tpl
22 svc-file-f90 file-f90.tpl
23 svc-file-f90-header file-f90-header.tpl
24 svc-file-f90-module file-f90-module.tpl
25 ...
26 /* services for actions */
27 svc-action-c-source action-c-source.tpl
28 ...
29 /* services for application tasks */
30 svc-app-chmod app-chmod.tpl
31 svc-app-copy-file app-copy-file.tpl
32 ...
```

19/41

FIG. 31

```
1 coll-macro-platform.tpl:
2 # This file defines platform-specific makefile macros
3
4 fragment-begin
5 _marker_ marker-htree copy
6 # The holding area for shared files and libraries
7 HTREE=/site/h
8 fragment-end
9
10 fragment-begin
11 _marker_ marker-macros1 copy
12 # makefile platform name, virtual platform name
13 MP=win98.plt
14 VP=win98
15 fragment-end
```

FIG. 32

```
1 coll-macro-site.tpl:
2 # This file defines site-specific makefile macros
3 fragment-begin
4 _marker_ marker-macros1 copy
5
6 # places where shared files go
7 SHARE_DIR=$(HTREE)\share
8
9 # places where web pages go
10 HOST_WEB=www.your_domain.com
11 ...
12 fragment-end
```

20/41

FIG. 33

```
1 coll-macro-toolnames.tpl:
2 # define macros for various program names
3 fragment-begin
4 _marker_ marker-macros1 copy
5
6 LS=ls
7 DIR=dir
8 RM=rm
9 CP=cp
10 ZIP=zip
11 UNZIP=unzip
12 CC=gcc
13 LIB=ld
14 RMDIR=rm
15 fragment-end
```

FIG. 34

```
1 coll-macro-compiler.tpl:
2 # This file defines compiler options
3 fragment-begin
4 _marker_ marker-macros1 copy
5
6 # default compiler options
7 OPT=
8 DEBUG=
9 # default linker options
10 LIBSPATH = $(HTREE)/$(MP)
11 LDFLAGS= -s
12 LPP= -L
13
14 fragment-end
```

21/41

FIG. 35

```
1 coll-macro-suffix.tpl:
2 # defines macros for file suffixes for this platform
3 fragment-begin
4 _marker_ marker-macros1 copy
5
6 # objects, executables, libraries, archives
7 O=.o
8 SO=.so
9 X=
10 L=.a
11 A=
12 AWKS=.awk
13 SEDS=.sed
14 LEXS=.l
15 YACS=.y
16 CLASS=.class
17 fragment-end
```

FIG. 36

```
1 coll-target-defaults.tpl:
2 # This file defines default makefile targets
3 fragment-begin
4 _marker_ marker-targets0 copy
5
6 # default targets used by all makefiles
7 default: build
8
9 all: build exports
10
11 build:
12
13 exports:
14 fragment-end
```

FIG. 37

```

0 /* fragment commands */
1 fragment-begin / fragment-end
2 _marker_ marker-name copy
3 _macro_ macro-name append value1 value2...
4 _target_ target-name add-deps dep1 dep2 ...
5 _target_ target-name copy
6 _target_ target-name copy-force

```

FIG. 38

```

1 base-template.tpl:
2
3 # marker-htree
4
5 # marker-macros1
6
7 # marker-targets0

```

23/41

FIG. 39

```
1 makefile.out:
2
3 # The holding area for shared files and libraries
4 HTREE=/site/h
5 # marker-htree
6
7 # makefile platform name, virtual platform name
8 MP=win98.plt
9 VP=win98
10
11 # places where shared files go
12 SHARE_DIR=$(HTREE)\share
13 ...
14 LS=ls
15 DIR=dir
16 ...
17 OPT=
18 DEBUG=
19 ...
20 O=.o
21 SO=.so
22 X=
23 ...
24 # marker-macos1
25
26 # default targets used by all makefiles
27 default: build
28
29 all: build exports
30
31 build:
32
33 exports:
34 # marker-targets0
```

24/41

FIG. 40

```
1 prod-prog-pi.tpl:
2 # Define platform-independent macros for programs
3
4 fragment-begin
5 _marker_ marker-macros1 copy
6 # Initialize these macros so they are defined.
7 ALL_OBJS__prod_=
8 OBJ_PI__prod_=
9 OBJ_F90__prod_=
10 OBJ_F90_MOD__prod_=
11
12 # create one macro to hold all objects
13 ALL_OBJS__prod_=$(OBJ_PI__prod_) \
14 $(OBJ_F90__prod_) $(OBJ_F90_MOD__prod_)
15
16 # add marker to anchor linker macro later
17 # marker-link-cmd
18 fragment-end
```

FIG. 41

```
1 prod-prog-os.tpl:
2 # Define operating system macros for programs
3
4 # Adds program name dependency to build target.
5 fragment-begin
6 _target_ build add_deps _mprod_$(X)
7 fragment-end
8
9 # Adds program name dependency to export target
10 fragment-begin
11 _target_ exports add_deps _mprod_$(X)
12 fragment-end
```



25/41

FIG. 42

```
1 prod-prog-pd.tpl:
2 # Define platform-dependent macros for programs
3
4 fragment-begin
5 _marker_ marker-macros1 copy
6 # default compiler flags for this platform
7 CCFLAGS1= -Wall -ansi -pipe -l.
8 CCFLAGS2= -l- -c
9 fragment-end
10
11 fragment-begin
12 _marker_ marker-link-cmd copy
13 # linker command for this platform
14 LDLIBS=
15 LD__prod_=${CC} -o _mprod_ _lib_dirs_ \
16 $(ALL_OBJS__prod_) _lib_names_
17 fragment-end
18
19 fragment-begin
20 # add link command to target for program product
21 _target_ _mprod_$(X) copy
22 $(LD__prod_) $(LDFLAGS)
23 $(CHMOD) 775 _mprod_$(X)
24 fragment-end
25
26 fragment-begin
27 # add object dependencies to product target
28 _target_ _mprod_$(X) add_deps $(OBJ_PI__prod_)
29 fragment-end
```

+

|    |                                     |                                  |
|----|-------------------------------------|----------------------------------|
| 1  | <code>_prod_</code>                 | name of product from cspec       |
| 2  | <code>_mprod_</code>                | name of product file on disk     |
| 3  | <code>_ptype_</code>                | product type of current product  |
| 4  | <code>_src_file_path_</code>        | source file pathname             |
| 5  | <code>_src_file_name_</code>        | source file filename             |
| 6  | <code>_src_file_name_no_suf_</code> | source filename with no suffix   |
| 7  | <code>_target_list_</code>          | list of makefile targets         |
| 8  | <code>_target_name_</code>          | name of current target           |
| 9  | <code>_deplist_</code>              | list of dependent targets        |
| 10 | <code>_incl_dirs_</code>            | list of include directories      |
| 11 | <code>_lib_dirs_</code>             | list of library directories      |
| 12 | <code>_lib_names_</code>            | list of library names            |
| 13 | <code>_zpln_</code>                 | parallel target number 01,02,etc |

[illegible]

27/41

FIG. 44

```
1 makefile.out:
2 ...
3 # Initialize these macros so they are defined.
4 ALL_OBJS_myprog=
5 OBJ_PI_myprog=
6 OBJ_F90_myprog=
7 OBJ_F90_MOD_myprog=
8
9 # create one macro to hold all objects
10 ALL_OBJS_myprog=$(OBJ_PI_myprog) \
11 $(OBJ_F90_myprog) $(OBJ_F90_MOD_myprog)
12
13 # marker-link-cmd
14
15 # marker-macros1
16
17 # default targets used by all makefiles
18 default: build
19
20 all: build exports
21
22 build: myprog
23
24 exports: myprog
25 # marker-targets0
```

28/41

FIG. 45

```
1 makefile.out:
2 ...
3 # Initialize these macros so they are defined.
4 ALL_OBJS_myprog=
5 OBJ_PI_myprog=
6 ...
7 # create one macro to hold all objects
8 ALL_OBJS_myprog=$(OBJ_PI_myprog) ...
9 ...
10 # linker command for this platform
11 LDLIBS=
12 LD_myprog=${CC} -o myprog $(LDLIBS) \
13 $(ALL_OBJS_myprog) $(lb)
14 # marker-link-cmd
15 ...
16 # default compiler flags for this platform
17 CCFLAGS1= -Wall -ansi -pipe -l.
18 CCFLAGS2= -l- -c
19 # marker-macos1
20 ...
21 build: myprog
22
23 exports: myprog
24 ...
25 # add link command to target for program product
26 myprog: $(OBJ_PI_myprog)
27 $(LD_myprog) $(LDFLAGS)
28 $(CHMOD) 775 myprog
29 # marker-targets0
```

29/41

FIG. 46

```
1 file-c-source.tpl:
2 # process files
3
4 # add current source file to top src file macro
5 fragment-begin
6 _macro_SRC_C append_src_file_path_
7 fragment-end
8
9 # add current source file to product source file macro
10 fragment-begin
11 _macro_SRC_C__prod_ append_src_file_path_
12 fragment-end
```

FIG. 47

```
1 action-c-source.tpl:
2 # process files
3
4 # add compilation command under C object targets.
5 fragment-begin
6 _target_target_name_$(O) copy
7 $(CC) $(OPT) $(DEBUG) $(CCFLAGS1) \
8 _incl_dirs_ $(CCFLAGS2) _src_file_path_
9 fragment-end
10
11 # add dependency list to C object target.
12 fragment-begin
13 _target_target_name_$(O) add_deps_deplist_
14 fragment-end
```

FIG. 48

```
1 makefile.out:
2 ...
3 SRC_C= ../s/gnulinux2/cmdline.c ...
4 ...
5 SRC_C_prod_= ../s/gnulinux2/cmdline.c ...
6 ...
7 # default compiler flags for this platform
8 CCFLAGS1= -Wall -ansi -pipe -l.
9 CCFLAGS2= -l- -c
10 # marker-macros1
11 ...
12 # default targets used by all makefiles
13 default: build
14
15 all: build exports
16
17 build: myprog
18
19 exports: myprog
20 ...
21 cmdline.o: ../s/pi/cmdline.h ../lib/pi/libfuns.h
22 $(CC) $(OPT) $(DEBUG) $(CCFLAGS1) \
23 _incl_dirs_ $(CCFLAGS2) ../s/gnulinux2/cmdline.c
24
25 ...
26 # marker-targets0
```

FIG. 49

|    |                |                                     |
|----|----------------|-------------------------------------|
| 1  | collection     | c-my-example                        |
| 2  | coll-type      | ct-program                          |
| 3  | coll-desc      | A fileset example                   |
| 4  | svc            | svc-coll-cleanup                    |
| 5  | end-collection |                                     |
| 6  | product        | myprog                              |
| 7  | prod-type      | pt-program                          |
| 8  | libs           | mylib                               |
| 9  | svc            | svc-app-copy-file myprog myprog.bak |
| 10 | end-product    |                                     |

FIG. 50

```
1 cspec:
2 ...

3 product myprog
4 prod-type pt-program
5 prod-desc A normal program binary executable.
6 end-product

7 product myprog-2
8 prod-type pt-shared-object
9 prod-desc A shared object program executable
10 replace-name myprog
11 end-product

12 _prod_ becomes cspec name myprog-so
13 _mprod_ becomes diskfile name myprog

14 # add link command to target for program product
15 _mprod_$(X):
16 $(LD__prod_) $(LDFLAGS_prod_)
17 $(CHMOD) 775 _mprod_$(X)

18 # link target for product myprog
19 myprog$(X):
20 $(LD_myprog) $(LDFLAGS_myprog)
21 $(CHMOD) 775 myprog$(X)

22 # link target for product myprog-so
23 myprog$(SO):
24 $(LD_myprog-2) $(LDFLAGS_myprog-2)
25 $(CHMOD) 775 myprog$(SO)
```



FIG. 51

```

1 product-build-order.tbl:
2 # define relative build order among products
3
4 pt-initial 10
5 pt-data 50
6 pt-library 100
7 pt-program 1000
8 pt-script 1000

```

FIG. 52

```

1 makefile.out:
2 ...
3 # dependent targets mylib and myprog appear in proper
4 # product build order, from left to right
5 #
6 build: mylib myprog
7
8 mylib:
9 ...
10 myprog:
11 ...

```

FIG. 53

```

1 file-build-order.tbl:
2 # define relative build order among file types
3
4 ft-resource 10
5 ft-precompiled-cpp 20
6 ft-c-source 50

```

FIG. 54

```

1 makefile.out:
2 ...
3 # dependent targets mylib and myprog appear in proper
4 # product build order, from left to right
5 #
6 build: mylib myprog
7
8 mylib:
9 ...
10 myprog: myresource.rc myprecompiled-header.o cmdline.o
11 ...

```

35/41

FIG. 55

```
1 dirs-include.lst:
2 dir/gnulinux2 /site/myteam/include/gnulinux2
3 dir/gnulinux2 /site/myteam/include/gnulinux
4 dir/gnulinux2 /site/include/gnulinux2
5 dir/gnulinux2 /site/include/gnulinux
```

FIG. 56

```
1 # suppose these are paths to example include files
2 /site/include/gnulinux2/external-incl-file.h
3 /site/myteam/include/gnulinux/team-incl.h

4 # include files matched by search rules, in order
5 /site/myteam/include/gnulinux/team-incl.h
6 /site/include/gnulinux2/external-incl-file.h

7 _incl_dirs_ = -I /site/myteam/include/gnulinux \
 ... -I /site/include/inux2

8 makefile.out:
9 ...
10 file1.o: ../s/file1.c
11 $(CC) $(OPT) $(DEBUG) $(CCFLAGS1) \
12 -I /site/myteam/include/gnulinux -I /site/include/inux2 \
13 $(CCFLAGS2) ../s/file1.c
```

FIG. 57

```

1 dirs-library.lst:
2 dir/gnulinux2 /site/myteam/lib/gnulinux2
3 dir/gnulinux2 /site/myteam/lib/gnulinux
4 dir/gnulinux2 /site/lib/gnulinux2
5 dir/gnulinux2 /site/lib/gnulinux

```

FIG. 58

```

1 # suppose these are paths to example libraries
2 /site/lib/gnulinux2/gnulinux-lib.a
3 /site/myteam/lib/gnulinux/team-lib.a

4 # libs matched by search rules, in order
5 /site/myteam/lib/gnulinux/team-lib.a
6 /site/lib/gnulinux2/gnulinux-lib.a

7 _lib_dirs = -L /site/myteam/lib/gnulinux -L /site/lib/inux2
8 _lib_names_ = -l team-lib.a gnulinux-lib.a

9 makefile.out:
10 ...
11 LD_mprog = $(LD) -L /site/myteam/lib/gnulinux \
12 ... -L /site/lib/gnulinux2 \
13 ... -l team-lib.a -l gnulinux-lib.a
14 ...
15 myprog$(X): ...
16 $(LD_mprog) ...

```

FIG. 59

|    |                       |           |          |        |       |
|----|-----------------------|-----------|----------|--------|-------|
| 1  | virtual-platform.tbl: |           |          |        |       |
| 2  | #                     |           |          |        |       |
| 3  | #                     | Specific  | Generic  | Family | Every |
| 4  | # Name                | OS        | OS       | OS     | OS    |
| 5  | #                     |           |          |        |       |
| 6  | gnulinux2.plt         | gnulinux2 | gnulinux | unix   | pi    |
| 7  | sol28.plt             | sol28     | sol      | unix   | pi    |
| 8  | win98.plt             | win98     | win9     | win    | pi    |
| 9  | win95.plt             | win95     | win9     | win    | pi    |
| 10 | winnt40.plt           | winnt40   | winnt    | win    | pi    |
| 11 | win2000.plt           | win2000   | winnt    | win    | pi    |

FIG. 60

- 1 # fragment search directories for win98 platform
- 2 fragments/win98
- 3 fragments/win9
- 4 fragments/win
- 5 fragments/pi
  
- 6 # fragment search directories for gnulinux 2 platform
- 7 fragments/gnulinux2
- 8 fragments/gnulinux
- 9 fragments/unix
- 10 fragments/pi

FIG. 61

|    |                |                            |               |
|----|----------------|----------------------------|---------------|
| 1  | collection     | c-my-example               |               |
| 2  | coll-type      | ct-program                 |               |
| 3  | coll-desc      | A fileset example          |               |
| 4  | end-collection |                            |               |
| 5  | product        | myprog                     |               |
| 6  | prod-type      | pt-program                 |               |
| 7  | libs/pi        | mylib                      |               |
| 8  | libs/gnulinux  | mylib myother-gnulinux-lib |               |
| 9  | svc/pi         | svc-prod-name              | svc arguments |
| 10 | svc/gnulinux   | svc-prod-name              | svc args      |
| 11 | svc/win98      | svc-prod-name              | svc args      |
| 12 | end-product    |                            |               |

FIG. 61

FIG. 62

```
1 makefile.out
2 ...
3 myprog: file-001.o file-002.o ... file-100.o
4 $(LD_mprog) ...

5 # GNU make parallelism with -jobs argument will compile
6 # 4 files at a time to build the myprog target
7 #
8 make -j 4 myprog

9 # without a parallel make tool, makefile targets must be
10 # generated to offer parallelism, as follows:
11 #
12 myprog: myprog-01 myprog-02 myprog-03 myprog-04
13 myprog-01: file-001.o file-002.o ... file-025.o
14 myprog-02: file-026.o file-027.o ... file-050.o
15 myprog-03: file-051.o file-052.o ... file-075.o
16 myprog-04: file-076.o file-077.o ... file-100.o

17 # now parallel commands can be issued against parallel targets
18 # running on multiple machines
19 on machine1: make myprog-01
20 on machine2: make myprog-02
21 ...
22 # running multiple windows on one machine
23 in shell window 1: make myprog-01
24 in shell window 2: make myprog-02
25 ...
26 # or running in the background on one machine
27 in shell window 1: make myprog-01 &
28 in shell window 1: make myprog-02 &
29 ...
```

FIG. 63

```

1 action-c-source.tpl:
2 # process files
3 ...
4 # this line adds the parallelism-specific object file macro to the
5 # "master" or "top level" object file macro.
6 fragment-begin
7 _macro_ OBJ_PI__prod_ append $(OBJ_PI__prod__zpln_)
8 fragment-end
9
10 # this line adds current object file to correct
11 # parallelism-specific object file macro
12 fragment-begin
13 _macro_ OBJ_PI__prod__zpln_ append _target_name_$(O)
14 fragment-end
15
16 # this line adds the parallelism-specific object file macro as a
17 # dependency of the parallelism-specific build target.
18 fragment-begin
19 _target_ build__zpln_ add_deps $(OBJ_PI__prod__zpln_)
20 fragment-end

```

FIG. 64

```

1 makefile.out:
2 ...
3 OBJ_PI_myprog = file-001.o file-002.o ... file-100.o
4 OBJ_PI_myprog_01 = file-001.o file-002.o ... file-025.o
5 OBJ_PI_myprog_02 = file-026.o file-027.o ... file-050.o
6 ...
7 build_01: $(OBJ_PI_myprog_01)
8 ...
9 build_02: $(OBJ_PI_myprog_02)
10 ...

```



FIG. 65

```
1 makefile.out:
2 # sequential and parallel targets for multiple products
3 ...
4 # target for building all products sequentially
5 build: build_01 build_02 build_03
6 ...
7 # parallel targets for building all products in parallel
8 build_01: myprog-01 product2-01 product3-01 ...
9 build_02: myprog-02 product2-02 product3-02 ...
10 ...
11 # target for building product 'myprog' sequentially
12 myprog: myprog-01 myprog-02 myprog-03
13 ...
14 # parallel targets for building product 'myprog' in parallel
15 myprog-01: $(OBJ_PI_myprog_01)
16 myprog-02: $(OBJ_PI_myprog_02)
17 ...
18 # target for building product 'product2' sequentially
19 product2: product2-01 product2-02 ...
20 ...
21 # parallel targets for building product 'product2' in parallel
22 product2-01: $(OBJ_PI_product2_01)
23 product2-02: $(OBJ_PI_product2_02)
24 ...
```